ECE8803-FML: DRSS Severity Classification on OCT images

Abhimanyu Rajeshkumar Bambhaniya
School of Electrical and Computer Engineering
Georgia Institute of Technology
Georgia, United States
abambhaniya3@gatech.edu

Yangyu Chen
School of Electrical and Computer Engineering
Georgia Institute of Technology
Georgia, United States
yangyuchen@gatech.edu

I. Introduction

The aim of this work is to perform Diabetic Retinopathy Severity Scale (DRSS) severity classification on Optical Coherence Tomography(OCT) images from OLIVES dataset [1].

What is the Motivation of this work? According to a 2004 medical survery [2], approximately 10.2 million US adults 40 years and older are known to suffer from diabetes mellitus (DM); 1 of every 12 person with DM in this age group has advanced, vision-threatening retinopathy. Diabetic retinopathy is a major cause of blindness, particularly among working-age adults.

How can we detect it? OCT is a medical imaging technique that provides high-resolution cross-sectional images of the retina, allowing for detailed visualization of retinal layers, fluid accumulation, and structural changes. OCT images for patients with diabetes provides valuable information about the presence and extent of retinal lesions and changes in retinal thickness, which can be correlated with the severity of diabetic retinopathy as assessed by DRSS [3].

Since the scale of this problem is humongous, We attempt to employ machine learning models, which can be trained on OCT images and can be used to classify the severity of diabetic retinopathy based on the presence and characteristics of retinal lesions, providing a **non-invasive**, **scalable** and **objective approach** for assessing disease severity.

We outline the content of our approach in the following sections. The flow of the paper is as follows: We start by analyzing the data provided in the Olives dataset. We elaborate on our analysis in section II. To solve this problem, we employ four ML approaches and train them on the part of the dataset. We discuss the details of their implementation in section III.

Next, section IV enlists the challenges we faced while training these models and the steps taken to mitigate those issues.

In section V, we show the effectiveness of the 4 methods in correctly classifying the DRSS. Finally, we conclude our findings in section VI.

II. Data Analysis and Curation

The data provide images for different DRSS severity that we can classify into three levels: 0 (DRSS levels 35 and 43), 1 (DRSS level 47 and 53), and 2 (DRSS level 61, 65, 71, 85).

Looking at the dataset, one can realize that each volume of data has the same DRSS severity level since the volume is a continuous series of frames for one patient visit. It makes sense to regard every volume as one entity for training and testing. Hence, we decided to stack images up based on their volume ID and train the stack as one. Hence, each datapoint is a volume of 49 layers of size 224x224.

III. Classification Methods

In this section, We describe the four methods we used to classify DRSS Severity in the OLIVES PRIME dataset.

A. 3D ResNet18

Due to the fact we are training based on volume data, we will need to use 3D ResNet18. We are able to find an open-source implementation of 3D ResNet18 for medical image segmentation [4]. We modified its last fully connected layer to fit our classification:

```
nn.Sequential(
    nn.Linear(512, 256),
    nn.ReLU(),
    nn.Dropout(p=dropout),
    nn.Linear(256, num_class))
```

We added ReLU and dropout so that overfitting can be constrained. With this setup, he number of trainable parameters are 33.336 million. One addition note on this is that, the image only has one channel. We expand it to three channels by repeating the channels, hence it fits ResNet18 architecture.

On the other hand, if we have metadata involved during training, the model will be a bit different. We have a multi-model structure implemented with 3D ResNet18 and MLP working together as shown in Figure 1. We will have 32 output features from ResNet18, and it then input to an MLP network along with metadata from an MLP network as shown in code block below. **Note that during testing, we are not using any metadata information.**

```
nn.Linear(512, 32)  # last layer from RN

nn.Sequential(      # MLP for metadata
    nn.Linear(num_meta, 4),
```

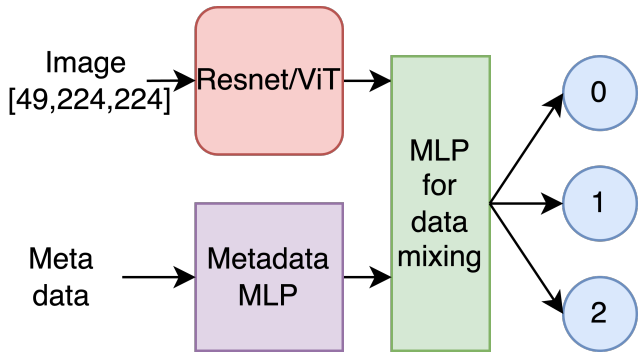


Fig. 1: Model layout for VIT and Resnet with metadata.

```
        nn.ReLU(inplace=True),
        nn.Linear(4, 4),
        nn.ReLU(inplace=True))

nn.Sequential(          # concatenate RN + MLP
        nn.Linear(32 + 4, 16),
        nn.ReLU(inplace=True),
        nn.Dropout(p=dropout),
        nn.Linear(16, 3),
        nn.Softmax(dim=1))
```

Overall, for ResNet18, We are using weighted cross entropy loss function, which will be discussed further in section IV. As for optimization, we use Adam with weight decay. Also, to make the regeneration of results easier, we fixed the seed to value 8803 during the training process. During training, we monitor the test dataset prediction after the completion of training every epoch, and hence capture the best result.

B. ViT

Like Resnet, we trained the ViT model on 3D volume images. To convert the 3D images to embedding inputs for ViT, we used a convolution filter with 49 filters and 16x16 kernel size. The output of this conv layer was the input to ViT, which was converted in embeddings with sequence length 196. ViT is a very large model compared to RN18, so we reduced the number of training parameters by changing the model size. Our implementation has three encoder layers, each with eight attention heads and an embedding dimension of 512. The MLP dimension was also reduced to 2048. The number of trainable parameters for this model is 15.999 million .

Next, the output of ViT was concate with the output of Metadata MLP through a few layers of MLP, and the final prediction of 3 classes was generated.

C. SVM

For SVM, we use the implementation available in sklearn with linear kernel. But the data feature dimension is too big to be fit on SVM. Hence, we reduce the dimension by applying HOG (Histogram of Oriented Gradients) and PCA (Principal component analysis) to the data. First, we did HOG, which basically divides the whole image into small cells, and for each cell, it calculates the gradient magnitude and orientation that can represent the direction of edges in that cell. After this, at the granularity of volume (49 images), PCA from sklearn will be applied and get the most important variations out. The default parameter, the number of components to keep for an image in PCA, is set to the smallest value from the number of samples and the number of features. Hence, in our case, it will be 49 (number of frames per volume). And again, we will be able to reduce the dimensions of the feature. The final dimension for a single volume is now reduced to 2401. At the end, the train data will be fed to fit the SVM model, and test data will be fed to get the prediction. The number of trainable parameters, in this case, is 1961.

D. KNN

We trained the model on a portion of the dataset. We concate the pixel value of the middle of the image volume. We use frames 10 to 39 and pixels between rows 106 to 224 and columns 80 to 450. **The main insight behind this was to focus on the more important pixel so they would have a higher correlation with DRSS.** Once data division was completed. We worked on raw pixel data to make the predictions.

IV. Challenges and Solutions

In this section, we will talk about what challenges we faced. Challenges can be coming from different models.

A. Limited Memory Space

Without doing anything, we feed the volumes into the network. We found that the memory space is not big enough to train the whole dataset simultaneously. Hence, we split both the training and testing datasets into batches.

B. Missing images

With the assumption that every volume will have 49 frames, we implemented the code to load the images by looping through all 49 indexes. However, we encountered an error during this process, and we realized that there were a few missing frames for a particular volume. So the solution was to fill that frame index with a neighbor frame on the fly. Hence, the missing frame's impact will be minimal since a volume is just a series of frames captured continuously.

C. Imbalance dataset

Now, with the batched dataset, we got the models running. And we can see the test balanced accuracy converge to 1/3 after some epochs. The predictions are full of 1, and that is not really useful. Looking at the data number, we realized that the high imbalance of data contributes a lot to this result. For the three classes, we have a distribution ratio of 0.32: 0.49: 0.19 roughly in both training and test set.

To balance the impact and contribution of these classes in our training process, we decided to use the weighted cross entropy loss function with the formula below. Hence, bigger weights are assigned to under-represented classes, classes 0

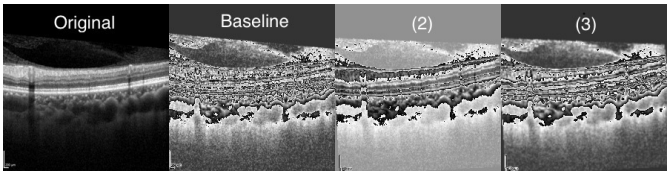


Fig. 2: Comparison between rotation-based augmentation for a class 0 image

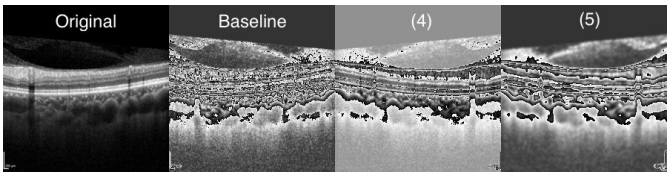


Fig. 3: Comparison between rotation-based augmentation for a class 0 image

and 2. Smaller weights are assigned to the over-represented class, class 1.

$$Weight[class] = \frac{\text{total\_num\_data}}{\text{num\_data[class]} \times \text{num\_classes}}$$

D. Small training set

The training dataset only contains 495 volumes, which is small for a deep model like 3D ResNet18 to learn. So, we introduced data augmentation to our training dataset. And such augmentation will be done on the fly, and no augmented data will be stored. We tried multiple kinds of augmentation. For example, gaussian blur, rotation, contrast, horizontal flip, and sharpness. And the final training dataset is composed of the following.

1) Baseline: Resize (224, 224) and Normalize
2) Baseline + Rotation by 3°+ Contrast with factor=0.5
3) Baseline + Rotation by 3°+ Gaussian Blur with kernel size=(5, 5)
4) Baseline + Horizontal Flip + Contrast with factor=0.5
5) Baseline + Horizontal Flip + Gaussian Blur with kernel size=(5, 5)

We decided on these based on a few factors: if it will contribute more diversity to the dataset, if it visually helps emphasize the edges, etc. An example of a class 0 image is shown in Figure 2 and Figure 3. We can see the contrast, or the gaussian blur is helping highlight the shape and edge at the central part of the image. And the rotation and horizontal flip add variation to the dataset. The number in captions on the images refers to the number in the data augmentation list above.

E. Overfitting on training set

We also encountered overfitting, and we applied a few methods to help reduce it:

- Dropout in fully connected layer
- Vary Batch Size
- Use AdamW, which adds weight decay regularization

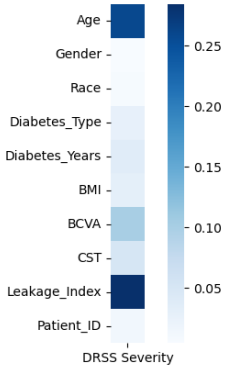


Fig. 4: Metadata Feature Correlation with DRSS Severity

F. Long training time

Training larger models like ViT took significant time; a single run would take 8 hours. To mitigate this issue, we implemented autoscaling, which can speed up train time on newer GPUs like V100 and A100 by switching from FP32, and FP16 to BF16 with a negligent impact on the model accuracy.

V. Evaluation

In this section, we will show various studies we conducted, their results, and what configurations are providing the best results. The best results for each classification is shown in Table I. And the phrase **best** refers to the **best-balanced accuracy** for the test dataset.

A. Detailed Analysis

1) 3D ResNet 18: The configuration, including hyper-parameters for training to get this result, is as follows Learning Rate(LR):1e-4, Model Dropout:0.5, Batch Size:16, Epochs: 20, Weight Decay:0.05, Metadata Feature used in training: Leakage_Index & Age. We also have data augmentation described in section IV. Also, we reiterate that we don't use any metadata during testing.

To get this configuration and understand how different factors contribute to the result, and how they impact the test-balanced accuracy we have, we conducted ablation studies by varying the configurations.

Table II shows the impact of LR on accuracy. we can see that at LR = 1e-4, the balanced accuracy performs the best. Especially for LR=1e-3, batch size of 4 and 8 has a balanced accuracy of 33.33%, which is the same as a coin flip in the context of three classes. This means the model is totally not learning from the training dataset. We can also see the impact of batch size here; with all other configuration fixes, we can see the batch size is not significantly impacting the result when the learning rate is appropriate. Otherwise, it impacts a lot as we can see at learning rate of 0.001 and 0.00001, batch size of 16 is doing much better than batch size of 4 and 8.

In an effort to explore the correlation between patient's clinical metadata and DRSS Severity by building a confusion matrix between metadata features. Figure 4 shows a portion of the confusion matrix that indicates the relationship between

TABLE I: Performance Matrices on four classifications

Classification	Trainable Parameters	Test Accuracy	Test Balanced Accuracy	Precision	Sensitivity	Specificity	F1
3D ResNet18	33.336 million	46.64%	**50.76%**	0.4945	0.5076	0.7454	0.5009
ViT	15.999 million	42.33%	40.66%	0.4066	0.4049	0.7056	0.4058
SVM	1961	41.10%	36.80%	0.3962	0.3680	0.6709	0.3816
K-NN	0	43.56%	40.29%	0.4029	0.3939	0.7188	0.3984

each metadata feature and DRSS Severity. We can see that *age* and *leakage index* have the strongest correlation. With this as the starting point, we experiment with how metadata features impact our results. We varied models to use the image only, two selected metadata features (leakage index and age), and all nine metadata features as shown in Table II. We can see that for all batch sizes, using two metadata features perform the best.

Similarly, We vary the dropout values in RN18 Table III; in this experiment setup, we exclude data augmentation, and use batch size of 4. We can see dropout of 0.5 perform better than the other two.

TABLE II: LR & # Metadata Features VS. Test Balanced Accuracy

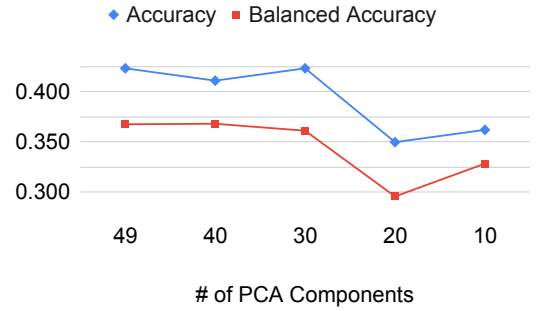
LR	# Metadata Features	Batch 4	Batch 8	Batch 16
1e-4	0	42.91%	46.04 %	37.76%
1e-4	2	50.09%	49.53%	**50.76%**
1e-4	9	44.94%	43.78%	45.79%
1e-3	2	33.33%	33.33%	44.93%
1e-4	2	50.09%	49.53%	**50.76%**
1e-5	2	48.26%	39.32%	38.95%

TABLE III: Dropout VS. Test Balanced Accuracy

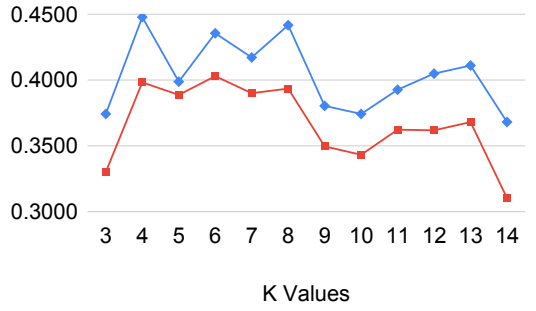
Dropout	Test Balanced Accuracy
0.1	40.66%
0.3	44.38%
0.5	**49.01%**

2) ViT: We trained the ViT model described subsection III-B in. We don't use any metadata for our best run. We ran 30 epochs with 3x the training data using augmentation. We also did ablation studies on ViT by varying the LR, Batch Size, metadata usage, and dataset size. In interest of brevity, we exclude those details from here and refer to experiments sheet in the dropbox.

3) SVM: For SVM, we conducted a series of experiment on the number of component to keep for PCA, and see how it impacts our accuracy. As shown in Figure 5. We can the accruacy overall increases when the number of components increases. And after the number of components is more than 30, the balanced accuracy trend tends to be flat and do no change much. Besides this, we also tried different kernels available in sklearn including poly, rbf, and sigmoid, but the accuracy results are similar to linear kernel and there are not significant difference.



(a) SVM Accuracy with varying Number of Component



(b) KNN Accuracy with varying K

Fig. 5: SVM & KNN Accuracy Evaluation

4) K-NN: For KNN, we observed the balanced accuracy trend when varying the number of nearest components. We vary 'K' from 3 to 14. The test balanced accuracy is 40% for K between three and eight. For lower and higher K values, the prediction accuracy drops.

VI. Conclusion

We used 4 ML techniques to predict DRSS severity based on OCT images of eyes. We get the best balanced prediction accuracy of 50.76% by using modified resnet18. We also find a strong correlation between patient age and leakage index to the DRSS severity.

Codebase: https://github.com/abhibambhaniya/DRSS-Severity-Classification-on-OCT-images

Presentation video and Spreadsheet link: Dropbox

References

[1] Mohit Prabhushankar, Kiran Kokilepersaud, Yash-yee Logan, Stephanie Trejo Corona, Ghassan AlRegib, & Charles Wykoff. (2022). OLIVES Dataset: Ophthalmic Labels for Investigating Visual Eye Semantics [Data set]. Advances in Neural Information Processing Systems 35 (NeurIPS 2022), New Orleans. Zenodo. https://doi.org/10.5281/zenodo.710523

[2] The Eye Diseases Prevalence Research Group*. The Prevalence of Diabetic Retinopathy Among Adults in the United States. Arch Ophthalmol. 2004;122(4):552–563. doi:10.1001/archopht.122.4.552

[3] Diabetic Retinopathy - Eye Disorders - Merck Manuals Professional Edition

[4] Ellis D.G., Aizenberg M.R. (2021) Trialing U-Net Training Modifications for Segmenting Gliomas Using Open Source Deep Learning Framework. In: Crimi A., Bakas S. (eds) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. BrainLes 2020. Lecture Notes in Computer Science, vol 12659. Springer, Cham. https://doi.org/10.1007/978-3-030-72087-2_4